Explainable Decision Support Systems with Semantic Tableaux and Defeasible Rules

Barbara Futyma, Eric Nakoja, David Pomerenke, Laurens Rutten, David Schimmel, JingYang Zeng
Supervised by: Dr. ir. ing. Nico Roos and Dr. Pieter Collins
Department of Data Science and Knowledge Engineering, Maastricht University

***Abstract*—Decision support systems as black box systems raise ethical and legal concerns as they, obstruct understanding and the attribution of responsibility. In an investigation of how to make decision support systems, an argumentation-based approach of creating explainable decision support systems is proposed. It makes use of non-monotonic reasoning, argumentation systems & defeasible rules to create an explainable decision tree from an input set of defeasible rules. The arguments and their support are provided as the explanation of how a decision comes about in the system. Two ways of implementing the argumentation logic are explored: The first based on semantic tableaux, the second based on backwards chaining of the rules. The developed systems are tested against example data sets from the legal domain.**

I. Introduction

A. Motivation

Decision support systems are a valuable help in improving the decisions-making process in various domains such as medicine and the legal system. However, they are often black box systems which means that they do not explain the reasoning behind their decisions. They provide valuable guidance in the face of complex situations where the facts and rules influencing a decision are hard to perceive by humans. But with the increasing usage of this technology in various fields, the relevance of the explainability of the decisions made by these systems is becoming clear, and a research community around explainable artificial intelligence has emerged. As decision support systems often work as black boxes, where the reasons for an outcome of the system are hard to comprehend, attention has been drawn to address them.

Black box decision support systems suffer from several problems stemming from the lack of understanding:

- Trust: If people do not understand for what reason a certain decision is being made, this may undermine the confidence that they have in that decision or the system responsible for making it. [8]
- Responsibility: It is at least questionable in how far a human is to be made responsible for making a decision based on a specialized automated decision support system, if that does not explicate its reasoning. If the reasons for a decision are transparent, it may be easier for a human decision maker to assume responsibility for such decision, as well as for external judges ascribing responsibility to them.

Therefore, it is an important goal to investigate ways of making the decision making process of automated decision support systems understandable.

1) Legal requirements on explainability: Decisions made by public organs in many states need to be explainable due to the requirement of the rule of law. Judges in the United Kingdom, France, and Germany are required to explain their decisions, and decisions may be rescinded on the sole basis of a lack of explanation. In the United States, this requirement has traditionally been more relaxed; it has been tightened recently in both the US and France. Not only judges but also administrative agencies are bound in many cases to explain their decisions, especially if they affect the rights of individual people. [8] This, in turn, calls for the explainability of automated reasoning. In the Netherlands, the *Rechtbank Den Haag* has recently ruled that an "[a]n administrative organ that partially bases its actions on [a system that a human user does not understand] is unable to properly justify its actions and to substantiate its decisions." [26] Moreover, the highest Dutch administrative court, *Raad van State* has recently decided that in order to apply automated decision systems for governmental decisions, they need to be fully comprehensible for judges as well as for normal citizens. [25]

Companies, in contrast, in principle need not justify their actions. The General Data Protection Regulation (GDPR) in the EU has recently introduced a *right to explanation* even towards companies [12]: In certain cases, especially related to personal profiling as used for the calculation of credit scores, clients are entitled to "obtain an explanation of the decision reached after such assessment and to challenge the decision". [33] This has been criticized as insufficient, and there is an ongoing debate about more advanced regulation concerning explainability. [10] The market for legal technology is substantial: While we are not aware of quantification of the global market, the investment in the innovation of legal technology is estimated at 260 million GBP in the United Kingdom in 2019. [28]

2) Explanation and argumentation: Artificial Intelligent systems are a promising application area for argumentation theory and its distribution. The inherently dialectical nature of argumentation provides a principled way in which structure exchange of, and reasoning about, arguments for claims between human and or automated systems engaged in collaborative information seeking, decision making, negotiation, persuasion dialogues and explanations.

A novel approach towards explainable artificial intelligence is to augment decision support systems, which are typically black box systems, with argumentation systems. The principle

idea behind this is that argumentation systems allow for the extraction of the reasoning leading to its conclusion. If the decision system can be mapped to the argumentation system in such a way that the reasons for the decision correspond to the propositions of the arguments, an explanation behind a decision can be extracted. To formally account for the possibility of exceptions from rules in decision making, the logic underlying the argumentation system can be extended by defeasible rules.

B. Problem Statement and Research Questions

Our problem statement is: Is it possible to create a decision support system that makes its reasoning understandable?

This project investigates the translation of formal specifications, mainly from the legal domain, into automated decision support systems. The focus is on creating a decision support system that makes the reasoning behind its decisions understandable. Two implementations of such system will be compared and evaluated on real world examples.

Research Questions:

- What is a suitable representation of information and arguments to allow for explainable decision support systems?
- How can we extract explanations for conclusions?
- How can the problem of redundant information/superfluous questions be avoided/dealt with?
- How can we explain the reasoning to the user?
- Is an argumentation based approach limited to formally restricted domains? If not, what needs to be changed to apply it to other domains?
- How well does an argument-based approach scale with respect to the number of rules involved?

II. Background Information/Preliminaries

A. Decision Support Systems

Decision support systems help users to solve decision problems, that is, binary classification problems. That means they output a recommendation that either confirms or rejects the proposition in regards to which the system is consulted by a human user. In the same way, the input should just conform or reject some propositions. This limits the application domain. It also makes the problems suitable for the logical reasoning approach followed in this report.

In a decision support system delivering explainable results, it becomes more necessary than it already is, not to ask the user questions that are not relevant to the outcome decision. That is because asking misleading questions can undermine the credibility of the system in the eyes of the user. Therefore, the decision support system should incorporate some means to ask the most relevant questions first. This may be done by considering the rules which can be applied in the decision making process and ask, for example, about the most general rules first. Then it can iterate and narrow in on the specific rules that apply to the particular scenario the user is in.

When considering the rules, especially in a binary representation, the question of exceptions also becomes relevant. For instance, when searching from the most general to the most specific rules as described above, rules may contradict or undermine each other. That might happen because some general rule is in use because it covers a lot of cases, and few exception cases are explicitly stated as to not increase the complexity of the rule set unnecessarily. It may also be the case that some specific rules lead to inconsistent outcomes, and they need to be evaluated specifically to see which rule applies (for example by consulting a rule or external source stating which rule should be applied in which case). Therefore, a mechanism of how rule disputes are to be resolved, should be provided.

The question of how to arrive at the rules that govern the decision support system is not of concern here. In the scope of the project, we assume that the rules can be extracted from the application domain under scrutiny and that way already make sense to the user of the system. Such would be the case especially in very formalized rule descriptions, like law texts. The more relevant question here becomes, then: How can the rules be extracted from such formal contexts and how can they be adequately represented and used for the decision making process and the construction of reasons.

Because decision support systems are interactive systems that ask the user for information whenever required, ideally the response time between interactions should not exceed a reasonable limit. What counts as reasonable may be up to debate. The primary concern in this project is not to assess the efficiency of the decision support system, but rather to provide a proof of concept that an explainable system is possible. However, as it is eventually an important criterion, for the sake of argument we assume an upper limit of two minutes to be an acceptable waiting time in-between interactions.

B. Explanations

Explainability is considered to consist of transparency (comprising simulatability, decomposability, and algorithmic transparency) and interpretability (comprising textual description, visualization, local explanations, and examples). [35] Analyses of various machine learning methods yield that the most suitable methods with regard to explainability are decision trees [3][35] and approaches based on deductive logic. [35]

Given that the decision support system devised here uses rules that are by themselves understandable (as they come directly from the application domain), the transparency condition is fulfilled if the algorithms determining how these rules affect the outcome are themselves transparent. The goal is, then, to provide interpretability by presenting the rules to the user in the way that they are relevant to the outcome. Because the rules themselves have a known textual description, showing the user what role they play in the production of the outcome will allow for an interpretation of the decision making process.

C. Argumentation systems

To make sense of the way how the rules relate to the problem to be decided by the decision support system, the notion of arguments is introduced. Arguments can be understood as the logical representation of how the facts and inference

rules relate to the problem. A formalization of arguments can be found in different kinds of argumentation systems. Argumentation systems are a popular tool for the analysis of complex arguments in artificial intelligence in the legal domain.

An argumentation system consists of inference rules, arguments, argument structures, and completeness conditions. The argument structure is the set that contains relating arguments. These arguments are connected by the inference rules in the form of a tree. The completeness conditions indicate the boundaries of the arguments.

An *abstract argumentation system* is defined as a set that contains some arguments and the expressed relations among the arguments. [34] Although researchers have developed various abstract argumentation systems [24], we start by considering abstract argumentation as proposed by Dung [9] for its generality. For our project, such an argumentation system would need to be innovated in terms of explainability and uncertainty.

Many Abstract argumentation systems emphasize accuracy and efficiency, however, interpretability is significant as well [36], because the comprehensive abstracting process and argument results can improve the credibility of decision.

D. First Order Logic

Argumentation systems are well explored for first order logic. Such a representation is also sensible in cases where all instances and rules can be sufficiently represented by a qualitative description, i.e. when there are no discrete metrics involved are can sufficiently be reduced to discrete classes. Given a decision support system guiding the user through a number of binary questions as described above, this is the case.

Because of the limited scope of the project, it is concerned only with propositional logic. This comes with the disadvantage that quantifying claims need to be explicated in separate propositions, making accounting for such claims more complicated. It also restricts how well some rules generalize, because all relevant properties need to be tailored every instance in which they apply. In contrast, abstracting away the properties from the entities they are associated with, as is done in predicate logic, allows for an easy generalization of these rules.

However, using predicate logic complicates the algorithms, which is why the project is limited to propositional logic. Since propositional logic is a subset of predicate logic, everything that can be expressed in the former can, in principle also be expressed in the latter. The proposed approach serves as a proof of concept that can be expanded by predicate logic in future work.

E. Non-monotonic reasoning & defeasible rules

As a way of handling exceptions to some rules is potentially required in the application domains specified above, the logics used in the approach should be able to account for that. Traditional propositional logic does not fulfill that condition. However, non-monotonic reasoning is a form of reasoning which is different from deductive reasoning where we can add nothing to change the conclusion. [27] Non-monotonic reasoning accepts new knowledge to negate the previous conclusion and draw a new conclusion. Therefore, non-monotonic reasoning helps the artificial intelligence system to deal with uncertain problems and special cases, when the intelligence system has incomplete or changing information. [2]

We use defeasible rules to account for the exceptions. Those can be defeated by other rules. When a rule is defeated than the conclusion drawn from it is no longer valid. We use the preference relation, as proposed by Roos [29], to define the hierarchy among the rules thus, the more preferred rule defeats the one with lower priority.

1) Defeasible propositions: For one of our implementations, we generalize the concept of defeasible rules and the concept of (non-defeasible) propositions to the concept of defeasible propositions. This has two advantages: *First,* via the preference relation, we can express degrees of uncertainty about facts, as well as about complex propositions that do not take the form of a rule. Moreover, we can keep track of explanations not only for the application of rules but also for the application about complex propositions. *Second,* the unified concept simplifies the implementation.

Equivalence of defeasible rules and (non-defeasible) propositions to defeasible propositions: We can convert a defeasible theory (a set Σ of (non-defeasible) propositions and a set D of defeasible rules) with a preference relation $<$ to a set of defeasible propositions D' and a preference relation $<'$ in the following way:

Forwards: We convert each defeasible rule $p \rightsquigarrow q$ with possibly complex antecedens p, possibly complex consequens q to a defeasible proposition by rewriting it as $p \rightarrow q$, and preserving its position in the preference ordering. We can convert a (non-defeasible) proposition p to a defeasible proposition by extending the weak preference relation on defeasible rules, such that p is strictly preferred to every defeasible rule:

$$D' = \Sigma \cup \{p \rightarrow q | p \rightsquigarrow q \in D\}$$

$$<' = \{(p \rightarrow q, r \rightarrow s) | (p \rightsquigarrow q, r \rightsquigarrow s) \in D\}$$

$$\cup \{(p \rightarrow q, t), | p \rightsquigarrow q \in D, t \in \Sigma\}$$

Backwards: We can convert a defeasible proposition p, to a defeasible rule by rewriting it as $True \rightarrow p$, or rewriting it as $p \rightarrow False$, and keeping its position in the preference ordering.

$$D = \{True \rightsquigarrow p | p \in D'\}$$

$$<= \{(True \rightsquigarrow p, True \rightsquigarrow q) | (p, q) \in <'\}$$

$$\Sigma = \emptyset$$

F. Theorem proving

Truth tables are unsuitable for theorem proving due to their strictly exponential complexity. There are four common methods for theorem proving: Sequent calculus [19], calculus of natural deduction [13], [19] (both used mostly by philosophers and other humans), resolution [32] (used mostly by computers), and semantic tableaux [19], [31] (used by both humans and computers). [30]

Semantic tableaux rely on a specific ruleset for the expansion of nodes, and this flexibility makes them applicable to many different kinds of logic. Rulesets have been devised for propositional logic, predicate logic, modal logic, dynamic logic, [30], doxastic logic, [30], epistemic logic, [30] and natural logic [1], among others. Recently, a version for defeasible logic has been proposed. [31] The expectation is that the tree-like structure of a semantic tableaux makes extracting the relevant reasons for why an argument holds or is refuted straight-forward.

III. Related Work

A. Decision support

Merritt [16] explain how to build an explainable decision support system based on Prolog. The key differences to our approach are that they use Prolog, which is a restricted version of predicate logic, whereas we use full propositional logic with the potential to extend it to full predicate logic; and that our approach specifically addresses defeasible rules and argumentation, which makes the application to law easier, and the explanations more understandable.

Zhong et al. [37] recently reported on the development of an explainable decision support system using assumption-based argumentation. For explanations, they generate an argumentation tree, like we do; and, as a unique feature, they also give an algorithm to generate natural-language explanations from this tree. They also give exemplary applications to the legal and medical domain.

Brarda et al. [5] most recently presented a prototype[1] for an explainable decision support system based on argumentation. They specifically addressed the issue of exceptions to rules by using conditional preference (*Cpref*) rules, that state which rule is preferred under which circumstances. A *Cpref* rule can arguably be imitated by our more general approach by using a logical implication in the consequent of a rule; a more detailed comparison, however, could be future work.

The existing product that is most similar to our approach is *Docassemble*[2], an open source decision support system for lawyers, that has its own small conference *Docacon*[3]. *Docassemble* carries out interviews with clients, and the lawyer is able to customize the content of the interviews using Python, YAML and Markdown. The interview logic is based on predefined questions that may have variables as a requirement, so that other questions about these variables have to be asked first.[4] This allows, in principle, for the creation of interviews that do not contain irrelevant questions, just like with our approach. However, the interview logic needs to be devised in part by the lawyer, and, with its procedural approach, is much less flexible than our descriptive approach based on propositional logic. *Docassemble* addresses explainability by allowing the user at every step to see the relevant underlying code of the interview logic written by the lawyer that leads to the question being asked.

On the commercial side, *Neota Logic*[5] is most notable for providing legal decision support systems, and their product has been used in the context of teaching computational law.[6] It is not transparent how their decision support system works internally. *Checkbox*[7] and *Afterpattern*[8] offer further legal decision support systems claiming to ask questions based on conditional logic. None of these commercial products explicitly address the issue of explainability.

B. Applications

For instance, Craven et al. [6] presented an assumption-based argumentation system and applied it in a medical domain where medical knowledge was derived from the results of clinical trials of early-stage breast cancer treatments and was represented by defeasible rules. They investigated if their system can find arguments whether a daily course of a specific drug for two years was justified for an example patient.

A common implementation of argumentation systems relies on the tableaux proof method. In that vein, Fox et al. [11] propose augmenting decision support systems with defeasible argumentation logic utilizing the ASPIC-framework as a framework for implementing schemas of how certain arguments can support certain actions or plans as a result of the decision support system. We want to expand on that idea and provide a concrete implementation of a decision support system augmented by an argumentation system utilizing defeasible logic in semantic tableaux proofs. However, we do not want to rely on an external framework like ASPIC.

C. ASPIC+

One framework for structured argumentation that is widely used is ASPIC+. [18] ASPIC+ allows its users to generate abstract argumentation frameworks in the sense of Dung. [9] It allows the user to freely choose the logical language, inference rules (strict and defeasible), knowledge bases (axioms and ordinary premises), and the argument preference ordering. When it is given these parameters it can tell the user how arguments can be built within the rules, how they can be attacked, and how these attacks can be resolved. This is very useful for this project since it allows the user to see how the system comes to its conclusion.

[1] https://github.com/buronbrarda/adds_with_cprefrules

[2] https://docassemble.org/

[3] https://docacon.com/

[4] Cf. https://docassemble.org/docs/interviews.html

[5] https://www.neotalogic.com/neota-logics-client-app-gallery/

[6] https://applications.neotalogic.com/a/links, https://curriculum.law.georgetown.edu/course-search/?keyword=LAWG+1040+01

[7] https://checkbox.ai/solutions/decision-support/

[8] https://info.afterpattern.com/product

The main limitation of ASPIC+ is that it is not a system but a framework used for specifying systems. The user still has to define the rules of the system and the reasoner, ASPIC+ only gives insight into the conclusions.

D. Reasoner implementations

The most prominent use of *resolution* is in the Prolog programming language. However, Prolog is technically limited in that it only supports Horn clauses rather than a full logic, and in that it uses depth-first search. [32] While there are extensions to meet both problems, the more fundamental problem in the context of this project is its lack of providing explanations. A Prolog-like logic programming language based on semantic tableaux called *Tablog* has been proposed, but no implementation could be found. [14] Moreover, resolution-based theorem provers for predicate logic have been developed in Python.[9][10]

Notable for *defeasible logic* is the *Oscar* system developed by Pollock, which has the high aim of creating a general rational agent. [20] It is implemented in LISP,[11] and uses natural deduction for propositonal and predicate logic, with an extension for defeasible logic. [21], [22] It is based on a variant of defeasible logic pioneered by Pollock, relying on the concepts or undercutting and rebutting defeaters. [20]

To our knowledge, there are no relevant publications regarding the implementation of *semantic tableaux*, but there are several open-source projects and libraries. For propositional logic, documented implementations are available in Haskell[12], Scala[13][14], and Javascript[15]. For predicate logic, there is a Prolog paper[23], a Haskell package[16] and an implementation in Javascript [17] (cf. [17]).

IV. Approach

The core of the investigated approach is: Using an analysis of arguments and their constituents, rules and propositions, to derive questions that are to be asked by the decision support system. These arguments, then, are also used to reconstruct the reasons for the eventual decision made by the system. So the explanation is dependent on the arguments and their analysis. Two Variants of that approach are considered here:

1) Semantic tableaux: The first approach uses semantic tableaux to determine which rules can be applied given the information present at each step of the decision making process. The argumentation system based on the semantic tableaux also generates all the arguments related to the main question.
2) Backward chaining: The second approach uses backwards chaining to construct a tree based inference rules form the final proposition of interest towards the the leaf nodes, which are are the farthest away from the final proposition in terms of inference steps while still related. Arguments are constructed by following the tree from the leaves to the root, and the search order is determined by a priority relation between amongst the inference rules. That priority relation corresponds, for the sake of the project, to the preference relation of defeasible rules, where applicable.

A. Tableaux Method

Forward construction of arguments: This variant utilizes the rather well understood argumentation frameworks for propositional logic. The semantic tableau method for defeasible rules is used to evaluate what kinds of rules can be applied given any sort of initial information. For the evaluation of whether or not the final proposition investigated holds or not, all relevant information needs to be present in the tableau. Then the analysis of the tableau also returns all the arguments in favour of the outcome. As stated above, however, the crux of a decision support system is that relevant information needs to be asked from the user. Therefore, this system needs to be supplemented. The chosen way is: to evaluate the semantic tableau as far as possible given the information it has at any given time. In this evaluation, rules are considered that have parts of their antecedence fulfilled, even if additional information is required to assess if it can actually be applied. As long as there are more of such candidate rules, they get applied or, if needed, a set of questions regarding information in the antecedence of the candidates is given to the decision support system. There, in turn, a heuristic for finding the most promising question can be applied and the question then be asked of the user. After the user inputs the new information, then the tableau gets re-evaluated.

In that iterative process, the information gets filled incrementally until no additional candidate rules remain. Finally, the tableau gets checked if it closes or not. Based on that, the arguments for or against the final conclusion are returned as the reasons for the final decision. The order in which questions get asked from the set of possible questions is determined by some heuristics. In an attempt to keep explanations as clean as possible and ask the most relevant questions first, this heuristics may correspond to the preference relation amongst the rules that may be applied with each given piece of information. The approach chosen in the prototype is to select that question first whose answer helps closing the most branches, so as to cover as many cases as possible. But more specialized heuristics are conceivable.

B. Argumentation tableaux

Semantic tableaux for propositional logic with defeasible rules are investigated as a way to construct decision support systems. This has the effect of both providing a grounds for reconstructing arguments in favour or against the decision

[9] https://github.com/yakuza8/first-order-predicate-logic-theorem-prover
[10] https://github.com/evhub/pyprover
[11] https://johnpollock.us/ftp/OSCAR-web-page/oscar.html
[12] https://github.com/ghulette/semantic-tableaux
[13] https://gist.github.com/anrizal06/1239945
[14] http://voidmainargs.blogspot.com/2011/09/semantic-tableaux-in-less-than-90-lines.html
[15] https://github.com/DACUS1995/Semantic-Tableaux-Method-for-Propositional-Logic
[16] http://hackage.haskell.org/package/tableaux, online demo: http://kashmir.dcc.fc.up.pt/cgi/tableaux.cgi
[17] https://github.com/wo/tpg

from a decision support system, and of providing a means of creating an efficient decision support system by leveraging the relations between the different rules from which the arguments are constructed.

The Argumentation tableau is implemented in two steps, searching for inconsistency and constructing arguments.

1) We initialize the add all the facts as well as the antecedences of the rules as tests, then expand the root node in order to search for an *inconsistency*.
2) We choose the arguments that have not been closed yet, to apply the applicable defeasible rules to generate new arguments. Subsequently, new arguments are added into the opening branches to check whether a new inconsistency is derived, see the Algorithm 2.

Then, the argumentation tableau tries to repeat 1) and 2) to expand the opening branches until it can not expand, the remaining opening branches are generated arguments.

Algorithm 1: search for inconsistency

Input: the basic information σ, the $root$ node of defeasible tableau
Output: a set of argument with the set of inconsistency IC

initialize: $root$ includes the set of σ and antecedence of R;
Start with $root$, expand each node:
for $\sigma_i,\sigma_j \in \{\sigma_1,\sigma_2,...,\sigma_n\}; j \neq i; \sigma_i,\sigma_j$ *are not decomposable* **do**
    if *not consistent*(σ_i,σ_j) **then**
        $newinconsistency$ = Argument($support$, $False$)
        if $newinconsistency$ *not in* σ **then**
            $\sigma = \sigma \cup$ newinconsistency
            node.children.add(σ)
for $\sigma_i,\sigma_j \in \{\sigma_1,\sigma_2,...,\sigma_n\}; j \neq i; \sigma_i,\sigma_j$ *are decomposable* **do**
    for $p \in decompose\_proposition(\sigma_i,\sigma_j)$ **do**
        $\sigma = \sigma \cup$ newinconsistency
        node.children.add(σ)
// expand child nodes

The argumentation tableau also provides an approach to evaluate arguments. When an inconsistency is derived, the weakest link or weakest last rule is defeated. For example, when we have an argument (1) and argument (2):

$$\frac{a \rightsquigarrow b}{d \vdash e \rightsquigarrow q} \vdash c \tag{1}$$

$$\frac{x \rightsquigarrow y}{m \rightsquigarrow n \vdash n \rightsquigarrow k} \vdash \neg c \tag{2}$$

Then, the defeasibility is calculated by the preferability of the defeasible rules. Subsequently, we evaluate the argument by its weakest link. If $a \rightsquigarrow b$ is the weakest link in argument (1), and $m \rightsquigarrow n$ is weakest link in the argument (2), and if the defeasible rule $a \rightsquigarrow b$ is weaker than $m \rightsquigarrow n$, the argument (2) is stronger than argument (1).

Algorithm 2: construct arguments

Input: the basic information σ, the $root$ node of defeasible tableau, and the set of defeasible rules R
Output: $pro\_argument$, $contra\_argument$

initialize $newargument$ = σ;
while $newargument$ *is not empty* **do**
    root.add($newargument$);
    IC = searchinginconsistency($root$);
    clear $newargument$;
    for $R_i \in \{R_1,R_2,...,R_n\}; i \leq n$ **do**
        if *antecedence of* R_i *matches* IC **then**
            $\sigma = \sigma \cup$ *conclusion of* R_i;
            $newargument$ = σ ;
    –
return $root$.argument;

Example IV.1. The "stand your ground law" allows people to use fatal forces to defend themselves when people believe they are in deadly danger [15], then this law makes the case of Michael Drejka [18] controversial. In this case, Michael Drejka fatally shot a man who shoved and knocked him to the ground, but Michael Drejka was acquitted.

According to the situation of Michael Drejka, we have basic information:

$$fact = \left\{ \begin{array}{c} t : (\{be\_threaten\}, be\_threaten), \\ a : (\{be\_attacked\}, be\_attacked), \\ k : (\{kill\_person\}, kill\_person), \\ b : (\{believe\_will\_be\_killed\}, \\ believe\_will\_be\_killed) \end{array} \right\}$$

Then, there are justifiable defense law and an consideration of murder.

$$defeasible\ rule = \left\{ \begin{array}{c} (t \wedge a \rightsquigarrow \\ (b \rightarrow justifiable\_defense)), \\ (k \rightsquigarrow conviction), \\ (justifiable\_defense \rightsquigarrow \\ \neg conviction) \end{array} \right\}$$

And we generate argument about the conviction by argumentation tableau.

$$\frac{k \rightsquigarrow conviction}{k} \vdash conviction$$

$$\frac{\begin{array}{c} b \\ t \wedge a \rightsquigarrow (b \rightarrow justifiable\_defense) \end{array} \vdash justifiable\_defense \quad \begin{array}{c} \\ justifiable\_defense \rightsquigarrow \neg conviction \end{array}}{\vdash \neg conviction}$$

Here we generated the two contrary arguments, both of them are answers of conclusion, then we can conclude the case by

[18] https://edition.cnn.com/2018/07/29/us/stand-your-ground-law-explainer-trnd/index.html

the priority of the defeasible rules, or generate new arguments by new add knowledge.

The argumentation tableau generates arguments with opening branches so that the current arguments still can be expanded by adding new knowledge. In other words, when the intelligent system is expected to draw a conclusion with an explanation, the argumentation tableau can generate all possible arguments as options, meanwhile, these arguments can be expanded continuously by new information. Therefore, the argumentation tableau is an ideal approach in Explainable Decision Support Systems.

a) Alternative Implementation: An alternative variant of the tableaux method may be construed: Instead of iteractively expanding the set of required propositions, an analysis of the tableau can be used to construct all possible scenarios as sets of propositions. A scenario refers to the configuration of truth values for all states of affairs that can potentially play a role in an argument. So the more propositions occur in the antecedence of rules while their truth value is not initially clear, the more combinations of proposition valuations constitute the set of configurations. Then, a heuristic can be applied to generate the propositions for information needed to realize any of these scenarios and thus should get asked by the decision support system as long as they are not already given.

This approach was rejected because of concerns regarding the exponential increase in computational complexity with growing sets of propositions for the scenarios. A potential downside of the iterative approach compared to the approach of surveying all possible scenarios is that the iterative approach might miss some of the arguments, depending on if the final proposition is tested for in its negated or in the positive form. If that is, indeed, a problem, a remedy to this is to apply the iterative method on both the positive and negated form of the proposition. But besides increasing the computational complexity by a factor, this can lead to additional problems. At least it should be made sure of that he decision support system does not ask irrelevant questions because of this, and a meaningful way of combining the arguments from both runs.

C. Searching for arguments

We pursue a second approach, where, -rather than building up all possible arguments from the available information and then checking whether they support the desired conclusion-, we instead construct all possible arguments (that is, *proto-arguments*) for the desired conclusion and then check whether they are supported by the available arguments. One could frame this approach as *backwards-chaining*, because the branches making up the arguments are constructed by applying rules or finding facts in order to support the conclusion, then finding rules or facts in order to support the previously found rules, and so on. The expected advantage of the search-based approach is that it will be able to deal with unknown information in a better way.

For this approach, we refer to a more general concept of *defeasible propositions*, rather than defeasible rules and (non-defeasible) facts. This concept, as well as converting a defeasible theory to it, is explained in subsubsection II-E1.

Algorithm 3: size

Input: proposition p
Output: size, that is, number of cases, of p
return $|\{Conj_1, ..., Conj_n\}|$
with DNF(p) $= \bigvee_i Conj_i$

Algorithm 4: unify

Input: proposition p, proposition q
Output: conjunction of consistent cases of p and q
return

$$\{Conj_{p,1}, ..., Conj_{p,n}, Conj_{q,1}, ..., Conj_{q,n}\}$$
$$\setminus (\{Conj_{p,i} | \forall i \neg\exists j : \neg(Conj_{p,i}, Conj_{q,j})\}$$
$$\cup \{Conj_{q,i} | \forall j \neg\exists i : \neg(Conj_{p,i}, Conj_{q,j})\}),$$
$$with \text{DNF}(p) = \bigvee_i Conj_{p,i}, \text{DNF}(q) = \bigvee_i Conj_{q,i}$$

Algorithm 5: arguments

Input: question proposition q,
set D' of rules and information
Output: set of possible arguments in favour of q

```
arguments = ∅
for d ∈ D' do
    restQuestion = unify(d, ¬q)
    isRelevant = size(restQuestion)
                 < (size(d) + size(¬q))
    isDecisive = size(restQuestion) > 0
    if isRelevant and isDecisive then
        arguments.add(p)
    else if isRelevant then
        arguments.add(
        pro = argumnents(¬restQuestion, D')
        contra = argumnents(restQuestion, D')
         ((pro, contra), d))
if size(q) = 1 and q is not the user question then
    arguments.add(Open({p_{1,1}, ..., p_{1,n}}))
    with DNF(q) = ⋁_i (⋀_{j=1,...,n} p_{i,j})
return arguments
```

1) Constructing proto-arguments: We construct the tree of possible arguments by running the *arguments* algorithm, once on the user question q, and once on its negation $\neg q$ to generate counter-arguments. Both times, the algorithm performs the following steps:

1) *Determining the relevant supports.* A proposition is relevant to the question, if at least one of its cases

Algorithm 6: questions

Input: argument a
Output: questions to be asked

```
if a = Open {p_1, ..., p_n} then
    return ⋀_{i=1,...,n} p_n
else if a = ((pro, contra), p) then
    P = {questions(p) | p ∈ pro, ¬defeated(p)}
    Q = {questions(q) | q ∈ contra, ¬defeated(q)}
    return ⋁(P ∪ Q)
else
    return ∅
```

(that is, one of conjunctive parts in disjunctive normal form, DNF) supports at least one of the cases of the question. This holds if one of the cases of the question is contradictory to one of the cases of the question. One way to check for it is to run the *unify* algorithm on the proposition and the negated question, and to see whether it removes some cases in comparison to the simple disjunction of the proposition and the negated question.

2) *Determining assumptions.* If the proposition supports all of the cases of the question, we can consider the proposition an assumption that needs no further support. This is the case if the *unify* algorithm on the proposition and the negated question returns an empty set of cases.
3) *Determining nested supports.* If the *unify* algorithm does not return an empty set of cases, we need some further support in addition to the proposition. Therefore, we create an argument containing the proposition, and search for propositions that support the cases that are still unsupported. We do this by running the whole *arguments* algorithm again, with the unification of the proposition and the negated question as a new input question to the algorithm. This recursively determines all possible arguments.
4) *Determining open supports.* If in the previous two steps no assumptions or nested supports for p could be found, we are in a hypothetical argument, which is not supported by the available information. In this case, we add a hypothetical pro support containing unify$(p, \neg q)$ and a hypothetical contra support containing $\neg$unify$(p, \neg q)$. We denote these arguments as Open$(cdot)$.

2) Resolving conflicts between rules: For each pro and contra argument created in the previous step, we check whether it is defeated by another argument. An argument is defeated if it is rebutted or undercut. An argument pro (contra) q is rebutted if it there is an argument contra (pro) q where the relevant proposition r is preferred (via the preference relation on propositions) to the relevant proposition in q, and if r is not undercut. A nested argument is undercut if all supports for it are rebutted or undercut. Applying these definitions to the arguments on the highest level will recursively determine all defeated and non-defeated supports in order to judge whether the arguments on the highest level are defeated or not.

3) Asking relevant questions: In order to determine the questions to be asked, we traverse the non-defeated branches of the tree and extract the disjunction of all open pro and contra supports at any level. If this disjunction is true, then some decision on at least one of the dominant possible argument can be made; either, it will be confirmed, or it will be discarded, and questions about the next most dominant possible argument can be asked.

From a user experience perspective, the problem arises of how to present these questions, appropriately representing the sufficient and necessary conditions implied by the extracted disjunction. A *first* approach would be to present multiple blocks of questions to the user, where the blocks correspond to the parts of the DNF of the extracted disjunction. Then, the user would need to answer all questions of one single block in order to determine whether one of the dominant arguments holds. We opt for a *second* approach and present to the user a block of questions corresponding to the parts of the conjunctive normal form (CNF) of the extracted disjunction; thus, they need to answer only one question in each block. This way, they can focus on each block one after another, consider which question they prefer to answer (for example, out of privacy concerns, or depending on how difficult it is to obtain the information in question), and then continue with the next block.

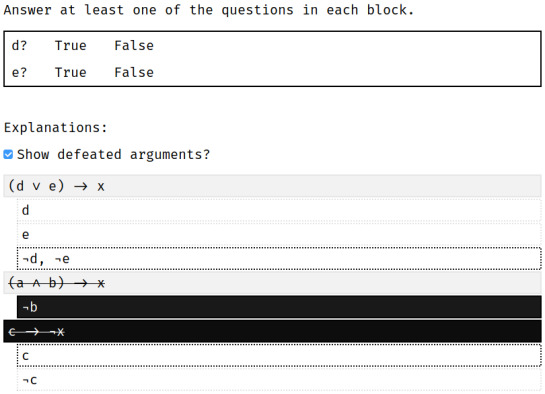

Fig. 1: Example of questions asked and explanations given. The input rules and facts for this example are (prefixed with priority) $3 : a \wedge b \rightarrow x$, $2 : d \vee e \rightarrow x$, $1 : c \rightarrow \neg x$, a, $\neg b$, and the user question is x.

4) Visualization of explanations: Figure 1 gives an example of how we visualize explanations: The pro arguments on the highest level are shown with a gray background, the contra arguments on the highest level are shown with a black background. Sub-arguments pro an argument are shown with the same background colour, sub-arguments against the parent argument are shown with the other background colour. Open assumptions (which the user still needs to validate or invalidate), are shown white boxes with dotted borders of the appropriate colour. Defeated arguments are striked through and hidden by default.

```
{
 "Tomas Cremers' Example": {
   "rules": {
     "1": {
       "antecedent": "EmployedAtCompany",
       "consequent": "CanMakeRequest"
     },
     "2": {
       "antecedent": "EmployedAtCompany &
           CompanyHasLessThanTenEmployees",
       "consequent": " CanMakeRequest "
     },
     "3": {
       "antecedence": "Employed &
           ReachedOldAgeInsurance",
       "consequent": " CanMakeRequestForChange "
     },
     "4": {
       "antecedence": "Employed & MilitaryOfficial",
       "consequent": " CanMakeRequestForChange "
     }
   },
   "facts": {}
  }
}
```

Fig. 2: Example of rules in JSON format

5) Intuitive explanations: One problem with this kind of visualization (and with the underlying or-tree representation used for arguments with this approach) is that the order of nesting of propositions within one branch (that is, one possible argument), is not always intuitive; furthermore, often there are multiple orderings possible. Thus, useful future work could be to investigate how with this second approach arguments of the same shape as in the first approach can be generated, as the shape of the arguments used there is more intuitive.

In order to construct such arguments, the question that is the input to the *arguments* algorithm, would first need to be decomposed into conjunctive normal form, and then all possible arguments for each disjunctive part of the conjunctive normal form could be searched. In this way, an and-or-tree could be constructed. We believe that such a tree would be more intuitive than the or-tree which we currently produce. This assumption does also needs further empirical investigation.

V. Experiments

A. Datasets

In order to evaluate the performance of the decision support system two main datasets are used; the first dataset is from the paper by Thomas Cremers. [7] This dataset focuses on rules related to the Dutch Flexible Working Act. The other dataset is comprized of the first two sections of the British Nationality Act. Both datasets allow for insight into the real world applications of this decision support system. To be able to use these datasets the rules are converted into a JSON format which can be read by the decision support system. An example of this JSON format can be seen in Figure 2.

B. Description

In Thomas Cremers' paper [7] some example scenarios are used. Firstly, we test a simple example consisting of 4 rules. Then we extend that example and evaluate the suggested approaches on a dataset consisting of 15 rules that may be relevant for deciding the outcome.

Bench-Capon, Coenen & Orton [4] show how to transform the first section of the British Nationality Act into logic rules. They also use some examples to show argument-based explanation can work. These examples are also used here.

The example from the British Nationality Act is if a person that is born in the United Kingdom and is older than ten years old. Who made a registration to be registered and who for the first ten years of his life has not left the United Kingdom for more than 90 days per year will be a British citizen. Altogether, the example used here is comprised of 12 rules that may need consideration to decide the outcome of the question of British Citizenship.

According to the paper about the British Nationality Act [4] the system should first consider whether the person is entitled to be registered. It should do this by checking if the person is born in the United Kingdom, is at least ten years old and for the first ten years of his live has not left the United Kingdom for more than 90 days per year. Using this is should determine that this person is entitled to be registered and because this person also made the application to be registered should be a British citizen.

In all the examples we do not give any initial information so that the system has to ask the user about all the relevant information.

C. Results

Cremers' test set did not yield results in over 5 minutes, failing the goal of having a responsive system providing interaction after at most two minutes. This was the case for either of the suggested approaches.

For the smaller example from the Cremers' paper (see figure 2 to see the rules involved), the first approach asks the following questions in this order:

1) Is 'LessThanTenEmployees' true? Yes
2) Is 'MilitaryOfficial' true? No
3) Is 'ReachedOldAgeInsurance' true? No
4) Is 'Employed' true? Yes

The system generates the following argument in favour of the conclusion:

*({({*Employed, LessThanTenEmployees*}, Employed* $\wedge$ *LessThanTenEmployees* $\rightsquigarrow$ $\neg$*CanMakeRequestForChange)}, $\neg$CanMakeRequestForChange)*

The second approach poses these questions in the following order:

1) Is 'Employed?' true? yes
2) Is 'LessThanTenEmployees' true? Yes
3) Is 'MilitaryOfficial' true? No Is 'ReachedOldAgeInsurance' true? No

The resulting argument generated by the system is: *(Employed* $\wedge$ *LessThanTenEmployees)* $\rightarrow$ $\neg$*CanMakeRequestForChange*
with the set of reasons: {*Employed, LessThanTenEmployees*}. In addition, the set of all refuted arguments in favour of this one can be be displayed.

British Nationality Act example, order of questions with the first approach:

1) Is 'ParentKnownNotBritish' true? No
2) Is 'EntitledToBeRegistered' true? Yes
3) Is 'BornAfterCommencement' true? No
4) Is 'ParentIsBritishCitizen' true? No
5) Is 'FoundAsNewbornInUK' true? No
6) Is 'FoundAfterCommencement' true? No
7) Is 'IsMinor' true? Yes
8) Is 'ForFirstTenYearsAbsentDaysLessThanNinety' true? Yes
9) Is 'ServingOutsideOfUK' true? No
10) Is 'ParentBritishCitizen' true? No
11) Is 'AdoptedUnderConventionAdoption' true? No
12) Is 'KnownToBeBornInNotUK' true? No
13) Is 'ApplicationToBeRegistered' true? Yes
14) Is 'AtLeastTenYearsOld' true? Yes
15) Is 'AdoptionAuthorizedByBritishCourt' true? No
16) Is 'SectionFiveRequirementsMet' true? No
17) Is 'ParentIsBritishCitizenNotByDescent' true? No
18) Is 'BornOutsideUK' true? No
19) Is 'AtLeastOneAdopterBritishCitizen' true? No

In this example it takes the system a long time to choose a question it wants to ask. Moreover, the time it takes to ask a new question increases the more questions the system asks.

The forward-chaining approach returns the following set of questions in the first iteration:

- 'AdoptedUnderConventionAdoption?'
- 'AdoptionAuthorizedByBritishCourt?'
- 'AllAdoptersHabituallyResidentInUK?'
- 'AtLeastOneAdopterBritishCitizen?'
- 'BornInUK?'
- 'BornOutsideUK?'
- 'FoundAfterCommencement?'
- 'FoundAsNewbornInUK?'
- 'KnownToBeBornInNotUK?'
- 'ParentKnownNotBritish?'

D. Discussion

Cremers' longer test set failing to be processed in reasonable time gives an indication of the problems that may be caused by runtime complexity. Running an additional test on the British national act revealed that the runtime becomes problematic with as few as 12 rules, where the decision support systems were still somewhat responsive, but the iterative approach took an increasingly long time to evaluate.

For the smaller examples, it could be shown that both systems can lead the decision-making process and provide the main argument line with both the rules and the support used in the rules. The second approach additionally can provide all the refuted arguments. Thus, someone who knows how the rules are encoded should also be able to understand the reasons for the decision.

The resulting arguments for the shown example are the same for both approaches and also conform to the common-sense reading of the rules and information used.

VI. Conclusion

Argument-based decision support systems such as the ones introduced here can be applied to a wide range of domains. The easiest application should be in domains in which the rules serving as the basis for the decision-making process are both codified and easily mapped to a human-understandable textual representation. A suggestion of how such a mapping can look like has been provided by the JSON example file, in which rules are identifiable by a number and the propositions constituting the rules are named to closely match their content in a brief manner. In order to make them applicable in rule-based arguments, any representation needs to make clear what the antecedence and the consequence of the rules would be.

The binary representation might make representing problems in which more granular values are important, as they would have to be represented by very specific rules. In that sense, domains in which this is the case, are unlikely to be coverable by the suggested approaches.

The codified rules are then used as the basis to construct arguments leading to the eventual decision. By keeping track of the arguments and possibly other arguments that were refuted on the way, they can be presented to the user as the reason and explanation of why the decision came about.

The problem of redundant information has been tackled implicitly, by using a heuristic covering as many cases as possible in the tableaux based approach and in the case of the backwards-chaining approach by checking at each iteration all the remaining rules whether they are rebutted are defeated in the light of the new information given after asking a question.

During the experimentation, it became apparent that the complexity of the search does not scale well for problems involving 12 or more rules.

The answer to the problem statement is thus: It is possible to create an explainable decision support system based on defeasible logic, and to successfully apply it to small examples from law.

A. Outlook

The feasibility of the proposed solution depends mainly on how well the rules can be encoded to be used in an argumentation-based approach. However, the efficiency concerns also play into this. In order to provide a plausible decision support system, the more rules that need to be considered, the more urgent that question becomes. Therefore, investigating ways of pruning the search space is warranted.

One potential way of improving the performance of the systems is revisiting how the normal forms are used in generating

sets of propositions the system would ask about. The formulation of the disjunctive normal form might be responsible for a drastic increase of complexity. However, in replacing it, it still needs to be assured that all potential arguments can be found, so the trade-off needs to be considered carefully.

As noted above, propositional logic might have difficulties in generalizing rules over multiple instance. Therefore, one step to develop the application further is to extend the systems to work with predicate logic.

If the mentioned problems can be resolved, argument-based decision support systems can play an important role in computer-assisted decision-making processes that allow for both, ascribing responsibility to the user of the decision making system and generating trust, if the reasons provided are agreeable by the user.

References

[1] L. Abzianidze, "LangPro: Natural Language Theorem Prover," *arXiv:1708.09417 [cs]*, Aug. 2017, arXiv: 1708.09417 version: 1. (visited on 09/26/2020).

[2] G. Antoniou and M.-A. Williams, *Nonmonotonicreasoning1997*. Mit Press, 1997.

[3] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," en, *Information Fusion*, vol. 58, pp. 82–115, Jun. 2020, ISSN: 1566-2535. DOI: 10.1016/j.inffus.2019.12.012.

[4] T. Bench-Capon, F. Coenen, and P. Orton, "Argument-based explanation of the british nationality act as a logic program," *Information and Communications Technology Law*, vol. 2, no. 1, pp. 53–66, 1993.

[5] M. E. Buron Brarda, L. Tamargo, and A. Garcia, "Using argumentation to obtain and explain results in a decision support system," *IEEE Intelligent Systems*, pp. 1–1, 2020. DOI: 10.1109/MIS.2020.3042740.

[6] R. Craven, F. Toni, C. Cadar, A. Hadad, and M. Williams, "Efficient argumentation for medical decision-making," in *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2012.

[7] T. Cremers, "Defeasible Logic as Professional Support for Regulation Analysis and Creation and Validation of the Specification of the Corresponding Services.,"

[8] F. Doshi-Velez, M. Kortz, R. Budish, C. Bavitz, S. Gershman, D. O'Brien, K. Scott, S. Schieber, J. Waldo, D. Weinberger, A. Weller, and A. Wood, "Accountability of AI Under the Law: The Role of Explanation," *arXiv:1711.01134 [cs, stat]*, Dec. 2019, arXiv: 1711.01134. (visited on 09/27/2020).

[9] P. M. Dung, "On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games," en, *Artificial Intelligence*, vol. 77, no. 2, pp. 321–357, Sep. 1995, ISSN: 00043702. DOI: 10.1016/0004-3702(94)00041-X.

[10] L. Edwards and M. Veale, "Enslaving the Algorithm: From a "Right to an Explanation" to a "Right to Better Decisions"?" *IEEE Security Privacy*, vol. 16, no. 3, pp. 46–54, May 2018, ISSN: 1558-4046. DOI: 10.1109/MSP.2018.2701152.

[11] J. Fox, D. Glasspool, D. Grecu, S. Modgil, M. South, and V. Patkar, "Argumentation-Based Inference and Decision Making–A Medical Perspective," *IEEE Intelligent Systems*, vol. 22, pp. 34–41, Nov. 2007. DOI: 10.1109/MIS.2007.102.

[12] B. Goodman and S. Flaxman, "European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation"," en, *AI Magazine*, vol. 38, no. 3, pp. 50–57, Oct. 2017, Number: 3, ISSN: 2371-9621. DOI: 10.1609/aimag.v38i3.2741.

[13] P. D. Magnus, *forall x. An Introduction to Formal Logic*, en, Version 1.4. [Online]. Available: https://github.com/OpenLogicProject/forallx/tree/fa620bb41d9fbb33dc2437878f0072a7c4dcbf51.

[14] Y. Malachi, Z. Manna, and R. Waldinger, "TABLOG: The deductive-tableau programming language," in *Proceedings of the 1984 ACM Symposium on LISP and functional programming*, ser. LFP '84, New York, NY, USA: Association for Computing Machinery, Aug. 1984, pp. 323–330, ISBN: 978-0-89791-142-9. DOI: 10.1145/800055.802049. (visited on 10/02/2020).

[15] C. B. McClellan and E. Tekin, "Stand your ground laws, homicides, and injuries," National Bureau of Economic Research, Tech. Rep., 2012.

[16] D. Merritt, *Building expert systems in Prolog*. Springer Science & Business Media, 2012.

[17] Ş. Minică, "RAESON: A Tool for Reasoning Tasks Driven by Interactive Visualization of Logical Structure," *arXiv:1507.03677 [cs]*, Jul. 2015, arXiv: 1507.03677. (visited on 09/26/2020).

[18] S. Modgil and H. Prakken, "The ASPIC + framework for structured argumentation: A tutorial," en, *Argument & Computation*, vol. 5, no. 1, pp. 31–62, Jan. 2014, Publisher: IOS Press, ISSN: 1946-2166. DOI: 10.1080/19462166.2013.869766. (visited on 09/10/2020).

[19] Open Logic Project, *The Open Logic Text*, Complete build, Revision 2451c5. [Online]. Available: https://github.com/OpenLogicProject/OpenLogic/tree/52451c5cfb55d621e395e0009253f4f6f505902a.

[20] J. L. Pollock, "OSCAR: A general-purpose defeasible reasoner," en, *Journal of Applied Non-Classical Logics*, vol. 6, no. 1, pp. 89–113, Jan. 1996, ISSN: 1166-3081, 1958-5780. DOI: 10.1080/11663081.1996.10510868. (visited on 10/02/2020).

[21] ——, "IV. Sentential reasoning," in *Oscar Manual*. [Online]. Available: https://johnpollock.us/ftp/OSCAR-

web-page/MANUAL/MANUAL-Chapter-4.pdf (visited on 10/02/2020).

[22] ——, "V. First-order reasoning," in *Oscar Manual*. [Online]. Available: https://johnpollock.us/ftp/OSCAR-web-page/MANUAL/MANUAL-Chapter-5.pdf (visited on 10/02/2020).

[23] J. Posegga and P. H. Schmitt, "Implementing semantic tableaux," in *Handbook of Tableau Methods*, Springer, 1999, pp. 581–629.

[24] H. Prakken, "An abstract framework for argumentation with structured arguments," en, *Argument & Computation*, vol. 1, no. 2, pp. 93–124, Jun. 2010, ISSN: 1946-2166, 1946-2174. DOI: 10.1080/19462160903564592.

[25] Raad van State, *ECLI:NL:RVS:2017:1259*, nl, Last Modified: 2020-03-03, May 2017. [Online]. Available: https://uitspraken.rechtspraak.nl/inziendocument?id=ECLI:NL:RVS:2017:1259 (visited on 09/09/2020).

[26] Rechtbank den Haag, *ECLI:NL:RBDHA:2020:1878 (English)*, nl, Last Modified: 2020-03-09, Mar. 2020. [Online]. Available: https://uitspraken.rechtspraak.nl/inziendocument?id=ECLI:NL:RBDHA:2020:1878 (visited on 10/02/2020).

[27] R. Reiter, "Nonmonotonicreasoning1998," in *Exploring artificial intelligence*, Elsevier, 1988, pp. 439–481.

[28] T. Reuters and L. Geek, *Legaltech Startup Report 2019—A Maturing Market*, en-GB, Section: Technology & Innovation, Oct. 2019. [Online]. Available: https://blogs.thomsonreuters.com/legal-uk/2019/10/18/a-new-report-legaltech-startup-report-2019-a-maturing-market/ (visited on 10/02/2020).

[29] N. Roos, "On Resolving Conflicts Between Arguments," *Computational Intelligence*, vol. 16, no. 3, pp. 469–497, 2000, ISSN: 1467-8640. DOI: 10.1111/0824-7935.00120. (visited on 10/06/2020).

[30] ——, "Logics for Artificial Intelligence," Lecture script, Maastricht University, Oct. 2019.

[31] ——, "A Semantic Tableau Method for Argument Construction," *BeNeLux AI Conference (BNAIC)*, 2020.

[32] S. J. Russell, P. Norvig, and E. Davis, *Artificial intelligence: a modern approach*, en, 3rd ed, ser. Prentice Hall series in artificial intelligence. Upper Saddle River: Prentice Hall, 2010, 37211, ISBN: 978-0-13-604259-4.

[33] N. Vollmer, *Recital 71 EU General Data Protection Regulation (EU-GDPR)*, en, text, Publisher: SecureDataService, May 2020. [Online]. Available: http://www.privacy-regulation.eu/en/recital-71-GDPR.htm (visited on 09/27/2020).

[34] G. A. W. Vreeswijk, "Abstract argumentation systems," en, *Artificial Intelligence*, vol. 90, no. 1, pp. 225–279, Feb. 1997, ISSN: 0004-3702. DOI: 10.1016/S0004-3702(96)00041-0.

[35] B. Waltl and R. Vogl, "Explainable artificial intelligence the new frontier in legal informatics," *Jusletter IT*, vol. 4, pp. 1–10, 2018.

[36] Z. Zeng, C. Miao, C. Leung, and C. J. Jih, "Building More Explainable Artificial Intelligence with Argumentation," en, p. 2,

[37] Q. Zhong, X. Fan, X. Luo, and F. Toni, "An explainable multi-attribute decision model based on argumentation," *Expert Systems with Applications*, vol. 117, pp. 42–61, 2019, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2018.09.038. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417418306158.